Performance Enhancement in solving Traveling Salesman Problem using Hybrid Genetic Algorithm

D. KAUR

Department of Electrical and Computer Science Engineering
University of Toledo
Toledo, OH_43606, USA

dkaur@eecs.utoledo.edu

M. M. MURUGAPPAN

Department of Electrical and Computer Science Engineering
University of Toledo
Toledo, OH_43606, USA

mmeyyapp@eng.utoledo.edu

***Abstract* - In this paper, a novel hybrid genetic algorithm for solving Traveling Salesman Problem (TSP) is presented based on the Nearest Neighbor heuristics and pure Genetic Algorithm (GA). The hybrid genetic algorithm exponentially derives higher quality solutions in relatively shorter time for hard combinatorial real world optimization problems such as Traveling Salesman Problem (TSP) than the pure GA. The hybrid algorithm outperformed the NN algorithm and the pure Genetic Algorithm taken separately. The hybrid genetic algorithm is designed and experimented against the pure GA and the convergence rate improved by more than 200% and the tour distance improved by 17.4% for 90 cities. These results indicate that the hybrid approach is promising and it can be used for various other optimization problems. This algorithm is also independent of the start city of travel whereas the result of NN algorithm are based on start city.**

I. Introduction

A hybrid genetic algorithm (or HGA) is a novel genetic algorithm based on some additional heuristics which improves the convergence rate of the algorithm as well as finds better solution. These HGAs are primarily used in complex optimization and search problems [7, 8, 9]. This paper discusses the HGA developed to solve the classical Traveling Salesman Problem (TSP) using the pure GA and the Nearest Neighbor (NN) heuristics [6]. Any genetic algorithm process works in stages and the different stages of a genetic algorithm are shown in the Fig. 1.

This paper is organized as follows. The section II explains the TSP in detail and also discusses the various methods used to solve TSP. The section *III* explains the pure GA for solving TSP and the various stages of the GA are explained in the sub-sections *IIIA* through *IIIG*. The section *IV* explains the design and implementation of the Hybrid GA. The section V summarizes the results from the simulations for pure GA and Hybrid GA. The section *VI* discusses the conclusions and the possible future works.

II. TRAVELING SALESMAN PROBLEM

The Traveling Salesman Problem (TSP) is a classical NP-hard Combinatorial Optimization (CO) problem in the field of computer science. Given n number of cities and the distances between each of the cities, the objective of TSP is to find the cheapest round-trip route that a salesman has to take by starting from any city and visiting all the cities exactly once and ending at the starting city. Euler introduced a form of traveling salesman problem in 1759, and it was formally named and introduced by the Rand Corporation in 1948. [1].

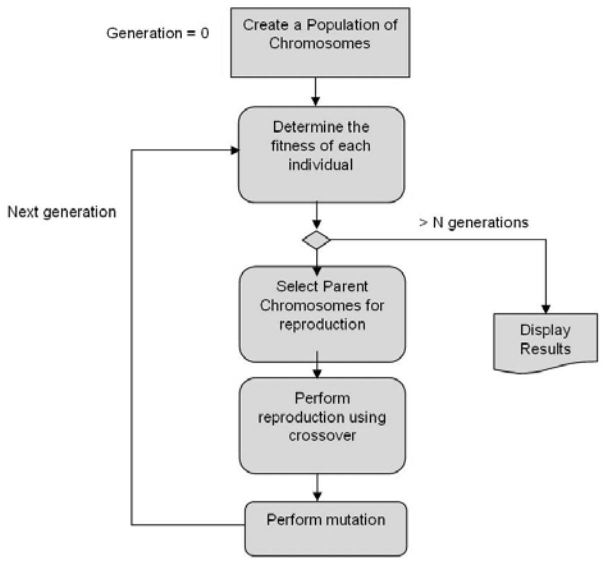
Generation = 0
Create a Population of Chromosomes
Determine the fitness of each individual
Next generation
> N generations
Select Parent Chromosomes for reproduction
Display Results
Perform reproduction using crossover
Perform mutation

Fig. 1: Stages of a Genetic Algorithm

A. Solving the TSP

The CO problems are tackled by many algorithms which are classified as either complete or approximate algorithms [1]. Solving the TSP has been an active area of research for a long time. The complete algorithm for solving TSP is to enumerate all possible paths and figure out which has the shortest distance. This turns out to be a bad idea due to the computational complexity. The usage of complete algorithm is limited for solving the TSP kind of problems.

The approximate algorithm is the best way to tackle the NP complete problems. Some of the approximate algorithms are Nearest Neighbor (NN) algorithm, Genetic Algorithm and Simulated Annealing [2].

III. PURE GENETIC ALGORITHM FOR TSP

The pure Genetic Algorithm for TSP involves various stages such as encoding, evaluation, crossover, mutation, elitism and decoding. These stages are explained in detail in the following section III A through III F.

A. Encoding of TSP

The encoding stage decides the format of the chromosome. A poor selection for the chromosome format might make the GA to get stuck up on a sub-optimal solution [4]. The usage of binary chromosome consisting of bits of 1's and 0's is not practical for TSP. This is due to the extremely large size of the chromosome and the complexity in the validation of solution after every generation. Even if all these difficulties are overcome, the convergence rate will be too low and the solution will not be optimal [4]. Thus a decimal chromosome is chosen for the GA-TSP. The genetic operations such as crossover and mutation can be defined easily since all the genes of chromosomes are integers from the domain {1, 2, n}. The maintenance of the validity of the chromosome is also simple, since all the genetic operations are done by manipulating genes (integers) and each gene corresponds to a city.

During the whole GA process, the two primary conditions to be met by a chromosome to be a valid solution are as follows.

- The length of the chromosome should be exactly equal to n.
- No integer in the range {1, 2, n} should occur more than once in a chromosome.

Each chromosome corresponds to a route and each gene corresponds to a city.

B. Evaluation of Chromosomes

The main goal of TSP is to minimize the tour distance and the same is used as the evaluation criterion. This evaluation function checks the fitness of a particular route. The lesser the distance traveled by the salesman in a route, the better the route is. The stopping criterion used here is just the number of generations evolved. Thus, the running of GA stops after completing certain number of iterations. The best chromosome in the last generation is the solution which may be either a local optimum or global optimum.

C. Crossover Operation

Crossover is the basic reproduction operation. The simplest form of crossover is used here. The evaluation function defined in section III.B assigns a fitness value for each chromosome. For every crossover operation, two chromosomes are randomly selected using roulette wheel selection. The chromosomes with higher fitness stand a better chance for getting selected for crossover. This selection of the chromosome with higher fitness for the crossover produces a better next generation with higher fitness values. The crossover operation continues until the specified crossover rate in met. The crossover rate for binary chromosomes is as high as 80-90%, whereas the crossover rate used here is 50% due to the decimal chromosomes [5].

Fig. 2 gives an illustration of the crossover operation for TSP of 8 cities. In this example, the parents selected for crossover are P1 as 46185327 and P2 as 32864715. For every crossover operation, 2 indices are chosen at random. The indices chosen for this example are 2 and 5, creating a window of cities in each of the chromosomes. The crossover operation exchanges these 2 windows (6185 and 2864) from each chromosome to the other first. The Initial Child1 (IC1) and Initial Child2 (IC2) chromosomes show this exchange of window. Then, the parent P1 is scanned gene by gene from the left end. In this example, the first gene scanned is '4'. Since '4' already exists in the window of IC1, the next gene is scanned. The next gene is '6', which also exists in the window of IC1. Then, the next gene is scanned and is '1'. This gene '1' does not exist in the window of IC1. Thus the gene '1' is inserted in the blank spaces of IC1 sequentially. This scanning of P1 continues until all the genes are scanned. This same operation is done for IC2 using the parent P2. The resulting two children Child1 and Child2 chromosomes are moved into the next generation population.

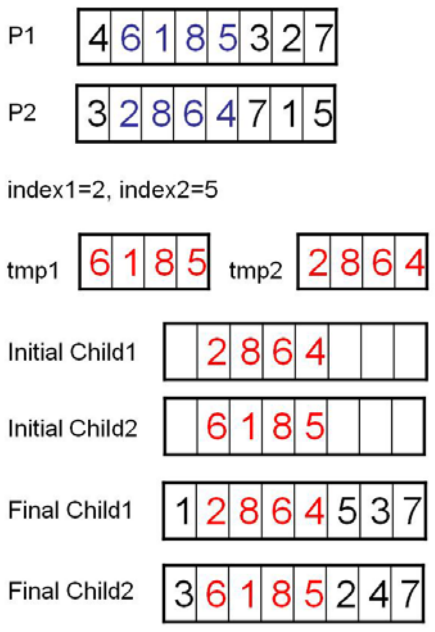

Fig. 2: Crossover in GA -TSP

D. Mutation Operation

The mutation operation in GA works on a single chromosome at a time and alters the genes of the chromosomes randomly. In this TSP application using GA, the mutation plays the primary role in arriving at the solutions, due to the usage of decimal chromosomes. Fig. 3 shows the example of mutation. The mutation operation is reversing the order of genes between the chosen indices. The GA chooses a particular chromosome at random for mutation. Two indices within the chromosome are randomly selected to create a window. The order of the genes with in the window is reversed. For a TSP of 8 cities, consider that the chromosome 36185247 is chosen for mutation. The indices chosen are 3 and 7 creating a window (18524) of cities in the chromosomes. The entire window of genes 18524 is completely reversed as 42581 to form the new chromosome 36425817.

This mutation operation on this particular Genetic Algorithm for TSP is very critical. The selection of a window mostly selects a sub-route which is optimized during the evolution of generations. Thus, by changing the position of the starting and end point of the sub-route, the solution keeps getting better. This way, the smaller sub-routes are generated using crossover and they are maintained and optimized using mutation.

The probability of mutation generally is very low of the order of 1 tenth of a percent for binary chromosomes. But, in case of decimal chromosomes, the mutation rate goes up to of the order of 85 %. This is due to the fact that, mutation produces the better offspring in the evolution and is the primary genetic operation in this case [5]. Sometimes, this mutation operation breaks the optimal sub-route also. In these cases, they are continuously eliminated by the mutation and crossover operations again over generations. If some better solutions evolve in the present generation, they are directly passed over to the next generation without any change by using elitism, which is the explained in section 3.6.

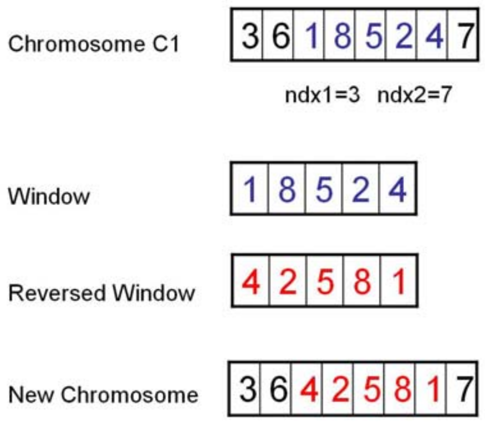

Fig. 3: Mutation in GA–TSP

E. Elitism

The elitism helps to keep the better solutions intact and pass over into the next generation without alteration. Elitism is always a helpful factor for any genetic algorithm and the rate of elitism should be optimal for the performance improvement. The elitism rate directly depends on the size of the population and the rate of elitism should be decreased when the population size is increased.

Here in the GA for TSP with the population size of 100, the elitism rate is set to 50%. Thus, the top 50 chromosomes in every generation are passed over to the next generation. The reason is because, in decimal chromosomes, the mutation plays the major part in the evolution of better solution. When trying to generate better solutions, the mutation operation also randomly worsens the best solutions found so far. Thus, to prevent this loss of better chromosomes, the elitism rate is set as high as 50%.

F. Decoding of Chromosomes

The decoding stage decodes the best chromosome in the final generation to the standard solution format. The decoding stage in the GA-TSP is simple due to the decimal representation of the chromosomes. After 500 generations are reached, the genetic algorithm comes to the termination. The best chromosome so far found is chosen and given as the solution. This best chromosome directly gives a decimal string and this is the route that the traveling salesman has to travel in order.

G. Simulation of Pure GA-TSP

The pure genetic algorithm for TSP is developed using the specifications mentioned in sections *IIIA* through *IIIF*. The number of cities is set to 90 and the position of these 90 cities are randomly generated and kept as a model for further comparisons. The Fig. 4 shows the location of the cities in a 10X10 grid. The pure GA approach is experimented and the results are shown in the Figs. 5 and 6.

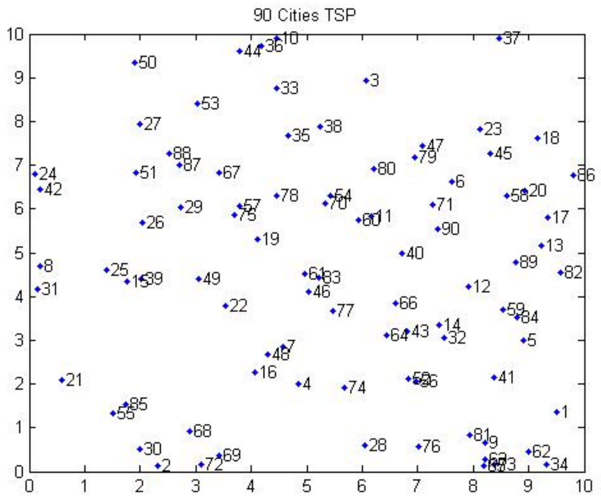

Fig. 4: Position of the cities in the 90 cities TSP

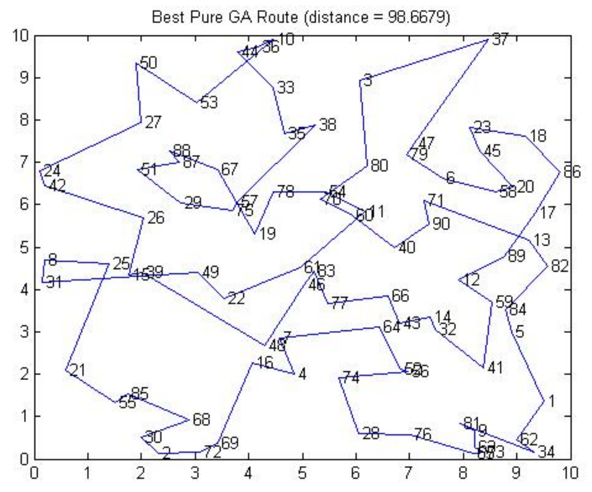

Fig. 5: Pure GA route for 90 cities TSP

IV. HYBRID GA FOR TSP

Hybrid Genetic Algorithms are used to improve the convergence rate and find more optimal solution over the pure GA [3]. The Hybrid GA designed here uses the Nearest Neighbor TSP heuristics for Initialization of population. Some of the other heuristics for TSP are Greedy heuristics, insertion heuristics etc. [5]. The NN algorithm is well established and is chosen to hybrid with GA to see the performance enhancement in solving TSP using Hybrid GA over the pure GA.

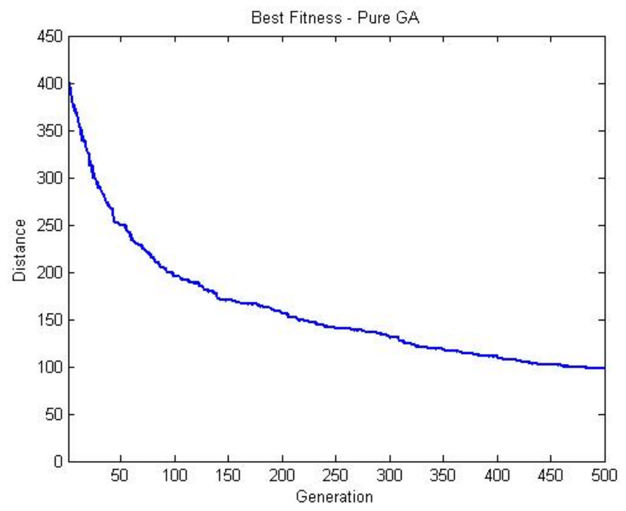

Fig. 6: Pure GA best solutions evolution over generations

A. Nearest Neighbor Algorithm:

The Nearest Neighbor (NN) algorithm is the most simple and straightforward heuristic for TSP. The algorithm generates the Nearest Neighbor routes for each city considering them as the starting city for that particular route. Nearest Neighbor algorithm is given below.

Step 1: Move all the cities to a list.

Step 2: Select the starting city as present city and remove it from the list.

Step 3: Find the nearest city to the present city in the list and make it present city and remove the present city from the list.

Step 4: Repeat step 3 until the list is empty.

Step 5: Return to the starting city and show the NN route corresponding to the starting city.

The complexity of NN algorithm is O (n2). The result of the NN algorithm also depends on the starting city chosen.

B. Nearest Neighbor Hybrid of GA

The NN algorithm is used initially for the TSP defined in section *III* and all the NN routes are found for each city as starting city. These NN routes are stored and analyzed for their fitness values. The better routes from this NN algorithm are introduced along with randomly generated solutions for the genetic algorithms. The percentage of pooling the solutions into the initial population is varied and the results are almost the same for both 10% filling and 100% filling. The percentage of filling the initial population with top routes found by NN is set to 10%. The remaining 90% of the population are generated randomly similar to pure GA. This gives the Hybrid GA to evade local optima and search for more optimal routes than the NN. This hybrid GA converges faster than pure GA and the results of NN algorithm and the Hybrid GA are shown in the figures 7, 8 and 9.

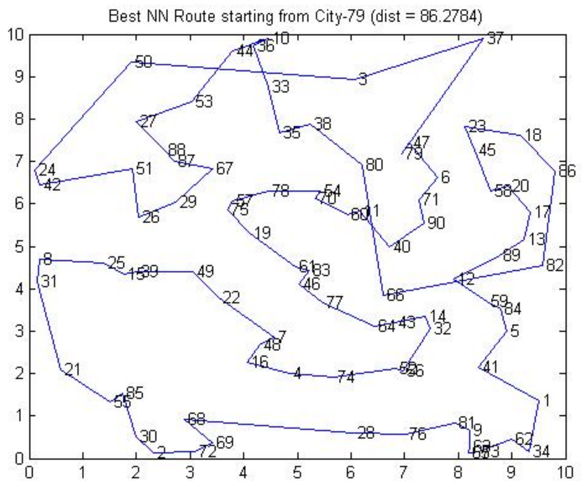

Fig. 7: Nearest Neighbor Route for 90 cities TSP

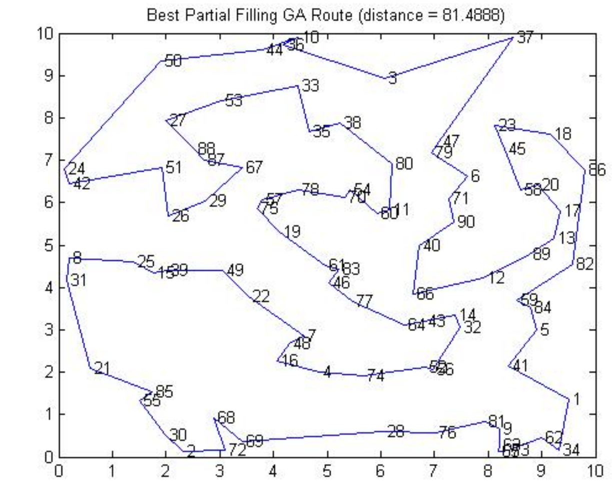

Fig. 8: Hybrid GA Route for 90 cities TSP

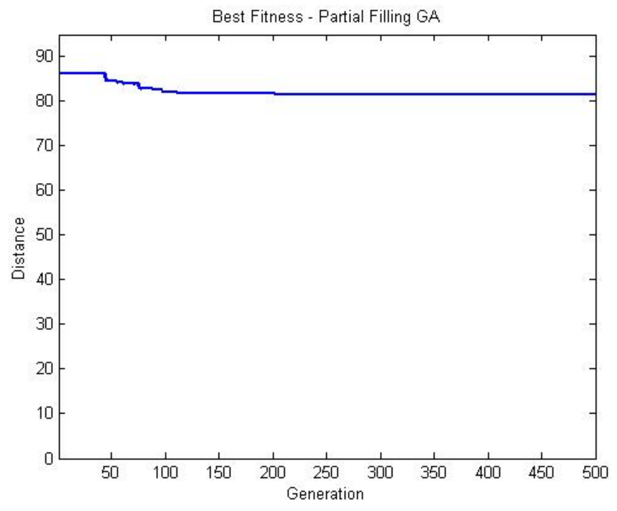

Fig. 9: Partial Filling GA best solutions evolution over generations

V. RESULTS

Table1 summarizes the results of the simulations for the varying number of cities using pure GA and the Hybrid GA. As seen from the Table 1, for 30 cities TSP, the pure GA converges in 150 generations at the tour distance of 43.7765, whereas the Hybrid GA converges in 40 generations at tour distance of 45.4814, which is slightly higher than pure GA. This is because, the number of cities is small, and making the Hybrid GA gets stuck in local optima. However, convergence speed improved by 325% in the Hybrid GA.

As the number of cities increases, both the tour distance and convergence rate improves in Hybrid GA over the pure GA. For 90 cities TSP, the convergence rate for pure GA is 500 and the tour distance is 98.67, whereas the convergence rate for Hybrid GA is 190 and the tour distance becomes 81.48. Thus, the convergence speed improves by 263% and the tour distance improves by 17.4 %.

TABLE 1

CONVERGENCE RATE FOR THE TOUR DISTANCE OF N CITIES BETWEEN PURE GA AND HYBRID GA

Cities N	Best NN Route Distance	Pure GA		Hybrid GA	
		Distance	Convergence Generation	Distance	Convergenc e Generation
30	48.1272	43.7765	150	45.4814	40
50	63.1816	61.9719	350	61.4153	20
70	73.7214	78.6589	500	69.9534	150
90	86.2784	98.6679	500	81.4888	190

Fig. 10 shows the tour distance vs. number of cities for the pure GA and Hybrid GA. It indicates that, the Hybrid GA is far better than the pure GA though it involves an additional complexity in determining the NN route. Moreover, the NN algorithm solution depends upon the starting city, whereas the Hybrid GA solutions are independent of the starting city.

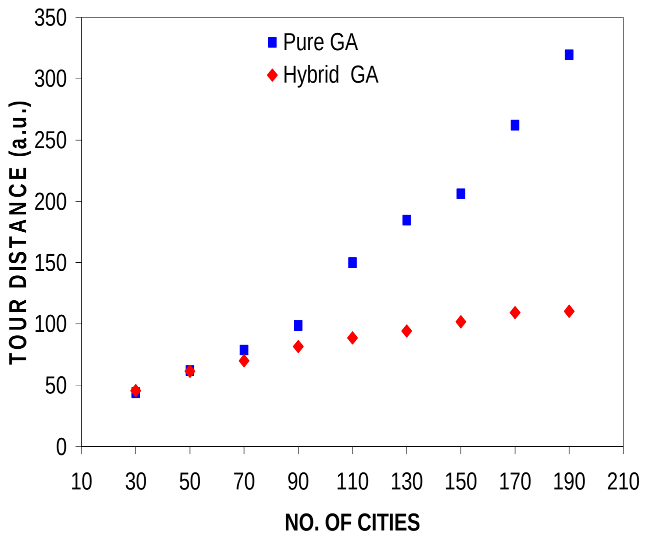

Fig. 10: Tour Distances of the Pure GA, Hybrid GA

VI. CONCLUSIONS AND FUTURE WORK

The importing of solutions from the NN algorithm into the initial population of the pure genetic algorithm gives faster and better convergence which can be seen from the results in Table 1. This hybrid approach also consumes lesser memory and lesser computational time. This way many of the complex combinatorial optimization problems can be transformed into the TSP format and worked on for better results. The representation of the chromosome and the genetic operations like crossover and mutation defined in this paper are specific to this implementation and these can be further modified and tuned for better results.

References:

[1] Randy L. Haupt and Sue Ellen Haupt, Practical Genetic Algorithms, John Wiley & Sons, 1998, ISBN: 0471188735.

[2] Chrisitan Nilsson. Heuristics for the Traveling Salesman problem. Linkoping University.
http://www.isid.ac.in/~dmishra/doc/htsp.pdf

[3] G. Andal Jayalakshmi, S. Sathiamoorthy, and R. Rajaram, A Hybrid Genetic Algorithm – A New Approach to Solve Traveling Salesman Problem, International Journal of Computational Engineering Science, Volume: 2 Issue: 2 p. 339 – 355.

[4] Steven L. Keast. A Simple Representation Technique to Improve GA Performance. Department of Computer Science and Software Engineering, Auburn University, Auburn, Alabama.
ftp://ftp.eng.auburn.edu/pub/techreports/csse/03/CSSE03-11.pdf

[5] John Yen, Reza Langari. Fuzzy Logic: Intelligence, Control and Information , Pearson Education. 2004. ISBN : 81-7808-906-8 Page: 472.

[6] Jie Gao, Mitsuo Gen, and Linyan Sun, A Hybrid of Genetic Algorithm and Bottleneck Shifting for Flexible Job Shop Scheduling Problem, GECCO'06, July 8-12, 2006, Seattle Washington, USA. ACM 1-59593-186-4/06/0007.

[7] Gregory Gutin and Abraham P. Punnen, The Traveling Salesman Problem and Its Variations by Springer, 2002. ISBN: 1402006640 Page: 9.

[8] Siddhartha Bhattacharyya. Evolutionary Algorithms in Data Mining: Multi-Objective Performance Modeling for Direct Marketing. KDD 2000, Boston, MA, USA. ACM 2000 1-58113 -233-6/00/08.